A new localization algorithm based on neural networks

Mathias Pelka*, Manfred Constapel*, Duc Tu Le Anh* and Horst Hellbrück*†
*Lübeck University of Applied Sciences, Germany
Department of Electrical Engineering and Computer Science
Email: mathias.pelka@fh-luebeck.de, manfred.constapel@fh-luebeck.de,
duc.tu.anh@stud.fh-luebeck.de, horst.hellbrueck@fh-luebeck.de
† University of Lübeck, Germany,
Institute of Telematics

***Abstract*—Indoor localization plays a major role in a wide range of applications. To determine the location of a tag, localization algorithm is required. In the past, machine learning algorithms were difficult to implement in consumer hardware, but with the advent of tensor processing units, even smartphones are capable to use artificial intelligence to solve complex problems. In this paper, we investigate a machine learning algorithm based on neural networks and compare the result to a linear least squares estimator. We design and evaluate different neural networks. Based on our observation, the neural network delivers poor performance compared to the linear least squares estimator.**

***Index Terms*—indoor localization, neural network, linear least squares.**

I. Introduction

A number of applications benefit from location information, e.g. in the industrial, medical or consumer sector. With location information, for instance, optimization in warehouses are possible. In a medical context, personal is guided directly to emergency situations. All applications require the computation of a location and different methods exists.

Precise localization is achieved using Ultra-wideband distance estimation, e.g. based on two-way ranging. Common hardware for implementation is the Decawave DW1000, a fully integrated single chip Ultra Wideband (UWB) transceiver IC, which enables precise timestamping of messages. Using those time stamps, distance estimation is possible which is in return used for location estimation.

Localization algorithms have been well investigated in the past, including linear least squares [1], Gauss-Newton iteration [2] or the Nelder-Mead algorithm [3]. In general, the accuracy and precision of a localization algorithm are in the magnitude of the measurement error [4].

Machine learning approaches have been studied in the past, e.g. Wymeersch et al. analyzed the received waveform in [5]. This approach requires analyzing the channel impulse response which is not always available. Another approach carried out by Savic et al. [6], employed kernel-based machine learning where selected channel parameters, are projected onto a nonlinear orthogonal space. Similar to the approach of Wymeersch et al. Savic et al. also employs the channel impulse response.

In the past, usage of artificial intelligence proofed resource intensive, but with the advent of new technologies, e.g. tensor processing unit (TPU), hardware specialized for machine learning [7], this is not a problem anymore. Furthermore, specialized software packages exist, to efficiently compute neural network even in resource-constrained devices [8].

In contrast to previous work and with respect to current results, we investigate neural networks to determine the location based on distance measurements.

The rest of the paper is structured as follows: Sec. II presents neural networks in general and Sec. III shows how we adapt a neural network for our problem. We present evaluation results in Sec. IV and conclude our paper in Sec. V.

II. Neural Networks

An artificial neural network (ANN) - or neural network (NN) for short - is a generic model used for prediction and classification tasks. At the minimum, it consists of one input layer and two neural layers, singly-linked from left to right: (1) input layer, (2) hidden layer, (3) output layer. Each neural layer is populated with at least one, but possibly a various number of neurons. Each neuron obtains inputs from the outputs of all neurons of the previous layer as illustrated in Fig. 1.

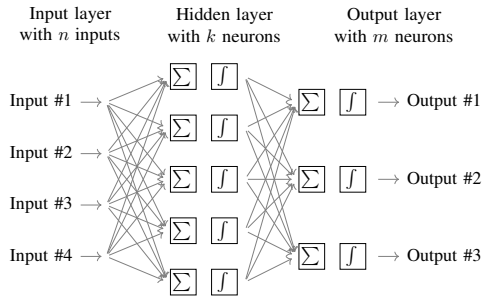

Fig. 1: Example Architecture of a 2-layer artificial neural network with $n = 4$, $k = 5$ and $m = 3$ neurons.

The inputs to a neuron are individually weighted. Subsequently, the weighted inputs are summed up, biased and fed to an activation function — usually, a step function approximated by a sigmoid function — in the figure indicated as the $\int$

https://doi.org/10.24355/dbbs.084-201807191240-0

symbol. Weights and biases of a neuron are estimated and optimized by backpropagation throughout a training phase. Backpropagation is a numerical method for minimizing the feedforward error within a neural network [9].

The number of hidden layers, as well as the number of neurons in each layer, is a design choice driven by heuristics. Since the problem of localization stated in our paper persists in a non-linear but straightforward-mapping from inputs (distances and positions) to outputs (locations), only a few hidden layers with a carefully chosen amounts of neurons are required for a decent training time while preventing overfitting. [10].

III. Neural Networks in indoor localization

The neural network in this paper consists of:

1) An input-layer with 12 input neurons, where eight neurons are used for receiving the x- and y coordinates of the reference points. The remaining four neurons are used for the distance measurements, between the tag and reference point.
2) Two hidden layers.
3) An output layer with 2 output neurons, representing the estimated x and y-coordinates.

In this configuration, the neural network supports four distance measurements. Since the tag has no inertial sensors, non-ranging or other features, e.g. IMU values, are not discussed in this paper. Nevertheless, the architecture can be easily adjusted to support more distance measurements and more output variables, e.g. to solve three-dimensional localization problems.

We also incorporate hidden layers into our neural network. To find the optimal number of hidden layers and neurons we systematically evaluate the results of different neural networks.

The weights of the neurons are randomly initialized with a standard-deviation following $2/\sqrt{x}$, where x is the number of the input values of the layer, as recommend by [9]. We use a linear activation function for the layers, which is a common choice for regression problems [9]. To evaluate the learning success of the neural network, we calculated the mean squared error between the input data and the output data. The neural network is implemented using the *TensorFlow* framework [8]. We trained the neural network over 100 000 epochs with a learning rate of $\varepsilon = 1 \cdot 10^{-5}$. The epochs describe how often we trained the neural network with our training dataset, while the learning rate describes the step size. If ε is too small, the learning takes very long, if it is too large, the network might not converge. For the training we assume a target area of 15×15 m and place four reference points at location $\mathbf{r}_1 = 2.5/2.5$, $\mathbf{r}_2 = 12.5/2.5$, $\mathbf{r}_3 = 2.5/12.5$ and $\mathbf{r}_4 = 12.5/12.5$.

One hidden layer, containing a finite number of neurons, can approximate every continuous function, [10], however, newer research [11] suggests that more hidden layers allow to reduce the number of neurons and to learn faster. Therefore we investigate different combinations of the number of hidden layers and the number of neurons in each layer to determine the optimal architecture. As of today a closed form solution for the optimal number of neurons and layers is not found [9].

We report the results of the training in Tab. I. The network is trained with four reference points, placed in the corner of a room with geometry 15×15 m. We calculate the distance to each grid point from each reference point and add noise based on a normal distribution with zero mean and $\sigma = 0.2$ m which we obtained from typical indoor distance measurements [12].

TABLE I: Mean square error (m) for learning rate $\varepsilon = 1 \cdot 10^{-5}$ and noise drawn from a normal distribution with zero mean and $\sigma = 0.2$ m.

Epochs		100	1 000	10 000	100 000	
Layers	Neurons					Time unit
1	5	34.5	11.95	1.34	0.20	0.64
1	10	53.4	14.08	0.90	0.20	0.75
1	15	49.8	7.64	0.28	0.20	0.99
2	5	68.9	8.11	0.44	0.20	0.69
2	10	92.2	2.28	0.29	0.20	1.00
2	15	9.18	0.42	0.20	0.20	1.55
3	5	33.03	6.54	0.54	0.20	0.87
3	10	60.9	1.80	0.20	0.20	1.28
3	15	10.7	1.16	0.20	0.20	2.57

Based on our investigation, we assume that after 100 000 epochs, all neural networks converge to a common mean squared error of 0.20 m. The common mean squared error indicates that neural network can't achieve better accuracy as already known algorithms. In general, the more hidden layers and the more neurons, the more training time is required for the network to converge. We normalized the runtime measurement to the neural network with 2 hidden layers and 10 neurons each. For the evaluation, we choose a network with 2 hidden layers and 10 neurons per hidden layer, which is visualized in Fig. 2. The figure shows only one input set, consisting of the x- and y-coordinate of the reference point and the distance measurement. The remaining configuration of the neural network is analogous.

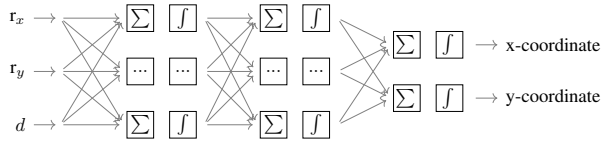

Fig. 2: Implemented neural network with 2 hidden layers. Only a subset of the input nodes is shown.

IV. Evaluation

For the evaluation, we assume the 15×15 m target area with the same location of the reference points as in the training. We divide the target area in a grid with an edge length of 0.25 m, resulting in a total of 3 600 data points. At each data point, we determine the true distances and added a Gaussian random variable with zero mean and standard deviation σ to simulate measurement inaccuracies. We then determine the location using the least squares algorithm and the neural network and calculate the Euclidean error between true location $\mathbf{r}$ and estimated location $\hat{\mathbf{r}}$. We repeated this 1 000 for each data point to retrieve reliable statistics, resulting in over 3 600 000 location calculations.

https://doi.org/10.24355/dbbs.084-201807191240-0

To evaluate the performance we calculate the mean, median, standard deviation, the interquartile range and the 95 percentile of the localization error $\mu = |\hat{\mathbf{r}} - \mathbf{r}|$ Furthermore, we determine the runtime of the neural network and linear least squares. We investigate the impact of the standard deviation for the performance. We present the results in Tab. II for least sqaures and Tab. III. We show an example visualization of the localization error of the neural network in Fig. 3. The figure shows symmetry caused by the location of the reference points.

TABLE II: Results for linear least squares for different σ.

σ (m)	0.10	0.20	0.40	1.00	2.00
mean error (m)	0.12	0.24	0.49	1.22	2.44
standard deviation (m)	0.03	0.05	0.10	0.25	0.51
median (m)	0.12	0.24	0.47	1.18	2.39
IQR (m)	0.03	0.07	0.14	0.35	0.69
95 percentile (m)	0.17	0.33	0.67	1.68	3.36

TABLE III: Results for the neural network for different σ.

σ (m)	0.10	0.20	0.40	1.00	2.00
mean error (m)	0.56	0.59	0.69	1.17	2.14
standard deviation (m)	0.31	0.30	0.26	0.21	0.27
median (m)	0.49	0.51	0.62	1.14	2.13
IQR (m)	0.38	0.36	0.29	0.23	0.35
95 percentile (m)	1.05	1.07	1.13	1.53	2.60

The mean runtime of the linear least squares is $0.08\,\mathrm{ms}$ where the mean runtime of the neural network is $0.21\,\mathrm{ms}$.

Based on our evaluation we conclude that linear least squares deliver better performance compared to the neural network. When the standard deviation σ increases over 1 m, the neural network delivers slightly better performance. The localization error of linear least squares increases roughly with the standard deviation, where the localization error of the neural network is at low noise levels almost constant.

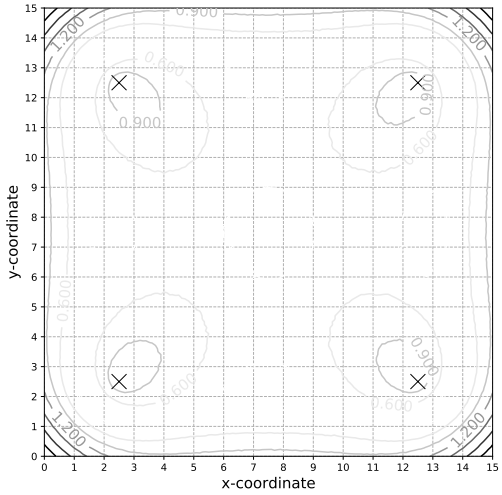

Fig. 3: Localization error of the neural network with a standard deviation of $\sigma = 0.2\,\mathrm{m}$. The crosses indicate the location of the four reference points.

The standard deviation of consumer-of-the-shelf hardware, i.e. based on the DW1000 from Decawave achieves a distance error of below 1 m. Consequently, we do not recommend neural networks for localization estimation, as linear least squares delivers better performance.

V. Conclusion and Future Work

In this paper, we have investigated neural network for location estimation based on distance measurements. We trained a neural network using the TensorFlow framework and designed it with two hidden layers. The input layer processes the location of the reference nodes as well as the distance measurement.

The network was able to estimate the location based on our evaluation, however, the performance at low noise levels was worse compared to linear least squares. When the distance measurement was affected by larger standard deviation $\sigma > 1\,\mathrm{m}$ of the noise component, the neural network delivers slightly better performance than linear least squares. Based on the evaluation we conclude, that our investigated neural network is not a suitable option to estimate the location.

Acknowledgments

This publication is a result of the research work of the Center of Excellence CoSA in the RosiE project (BMWi FKZ ZF4186102ED6). Horst Hellbrück is adjunct professor at the Institute of Telematics of University of Lübeck.

References

[1] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2007.

[2] G. Shen, R. Zetik, and R. S. Thoma, "Performance comparison of toa and tdoa based location estimation algorithms in los environment," in *Positioning, Navigation and Communication, 2008. WPNC 2008. 5th Workshop on*. IEEE, 2008, pp. 71–78.

[3] M. Pelka, P. Bartmann, S. Leugner, and H. Hellbrück, "Minimizing indoor localization errors for non-line-of-sight propagation," in *International Conference on Localization and GNSS*. IEEE, 2018.

[4] Y. Zhao, X. Fan, C.-Z. Xu, and X. Li, "Er-crlb: An extended recursive cramér–rao lower bound fundamental analysis method for indoor localization systems," *IEEE Transactions on Vehicular Technology*, 2017.

[5] H. Wymeersch, S. Maranò, W. M. Gifford, and M. Z. Win, "A machine learning approach to ranging error mitigation for uwb localization," *IEEE Transactions on Communications*, 2012.

[6] V. Savic, E. G. Larsson, J. Ferrer-Coll, and P. Stenumgaard, "Kernel methods for accurate uwb-based ranging with reduced complexity," *IEEE Transactions on Wireless Communications*, 2016.

[7] TechRadar, "Google's tensor processing unit explained: this is what the future of computing looks like," 2017. [Online]. Available: https://www.techradar.com/news/computing-components/processors/google-s-tensor-processing-unit-explained-this-is-what-the-future-of-computing-looks-like-1326915

[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, 2016.

[9] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc.", 2017.

[10] G. Gybenko, "Approximation by superposition of sigmoidal functions," *Mathematics of Control, Signals and Systems*, 1989.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, 2006.

[12] M. Pelka, D. Amann, M. Cimdins, and H. Hellbrück, "Evaluation of time-based ranging methods: Does the choice matter?" in *14th Workshop on Positioning, Navigation and Communication*. IEEE, 2017.

https://doi.org/10.24355/dbbs.084-201807191240-0